17

JC662 U.S. PTO 07/26/99

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	2130
First Inventor or Application Identifier	CABRERA et al.
Title	Extensible System Recovery Architecture
Express Mail Label No.	EJ660771087US

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

JC542 U.S. PTO 09/361641 07/26/99

1. [ ] * Fee Transmittal Form *(e.g., PTO/SB/17)* *(Submit an original and a duplicate for fee processing)*
2. [X] Specification [*Total Pages* 41] *(preferred arrangement set forth below)*
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. [X] Drawing(s) *(35 U.S.C. 113)* [*Total Sheets* 10]
4. Oath or Declaration [*Total Pages*]
   a. [ ] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 C.F.R. § 1.63(d)) *(for continuation/divisional with Box 16 completed)*
      i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

****NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).***

5. [ ] Microfiche Computer Program *(Appendix)*
6. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. [ ] Computer Readable Copy
   b. [ ] Paper Copy (identical to computer copy)
   c. [ ] Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. [ ] Assignment Papers (cover sheet & document(s))
8. [ ] 37 C.F.R.§3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
9. [ ] English Translation Document *(if applicable)*
10. [ ] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [X] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
13. [ ] * Small Entity Statement(s) *(PTO/SB/09-12)* [ ] Statement filed in prior application, Status still proper and desired
14. [ ] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
15. [ ] Other: ......

16. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment:*
[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: ________/__________________
Prior application information: *Examiner* ______________________ *Group / Art Unit:* ______________________
For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[ ] *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or [X] *Correspondence address below*

Name	Albert S. Michalik, Reg. No. 37,395				
	Michalik & Wylie, PLLC				
Address	14645 Bel-Red Road				
	Suite 103				
City	Bellevue	*State*	Washington	*Zip Code*	98007
Country	USA	*Telephone*	(425) 653-3520	*Fax*	(425) 836-3003

Name (Print/Type)	Albert S. Michalik	*Registration No. (Attorney/Agent)*	37,395
Signature	Albert S Michalik	*Date*	7/26/99

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

In re Application of CABRERA et al.
Attorney Docket No. 2130

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

"Express Mail" mailing label number EJ660771087US

Date of Deposit: July 26, 1999

I hereby certify that the following documents:

New Patent Application in the name of Luis Felipe Cabrera, Kartik N. Raghavan and Glenn A. Thompson for "Extensible System Recovery Architecture," including 1 Page Cover Sheet, 29 Pages Specification, 10 Pages Claims, 1 Page Abstract, 10 Sheets Of Drawings, Transmittal Sheet

are being deposited with the United States Postal Service "Express Mail Post Office To Addressee" Service under 37 C.F.R. 1.10 on the date indicated above and is addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231.

Albert S. Michalik
(Typed or printed name of person mailing paper or fee)

Albert S. Michalik
(Signature of person mailing paper or fee)

2130 express mail cert

S P E C I F I C A T I O N

TO ALL WHOM IT MAY CONCERN:

Be it known that we, Luis Felipe Cabrera, a citizen of the United States, residing at 2009 Killarney Way S.E., Bellevue, Washington 98004, Kartik N. Raghavan, a citizen of the United States, residing at 745 Summit Ave E., Unit 401, Seattle, Washington 98102 and Glenn A. Thompson a citizen of the United States, residing at 4223 205th Place NE, Redmond, Washington 98053 have invented a certain new and useful EXTENSIBLE SYSTEM RECOVERY ARCHITECTURE of which the following is a specification.

EXTENSIBLE SYSTEM RECOVERY ARCHITECTURE

FIELD OF THE INVENTION

The present invention is directed generally to computer systems, and more particularly to recovering from the failure of a computer system.

BACKGROUND OF THE INVENTION

Computer systems are protected against failure by backing up the computer data, whereby if a system crashes, the data may be restored. However, if a computer system fails in a manner in which it cannot reboot, the data cannot be simply restored. For example, hardware may fail, (e.g., the hard disk controller burns out), or software may fail, (e.g., a virus corrupts some key files and/or data), in a manner that prevents a reboot. However, in the event of a system failure, computer users not only want their data restored, but want their system restored to the way it was prior to the failure.

At present, backing up system information so as to enable a system to be restored to a bootable state, involves the use of many disjoint and separate programs and operations. For example, a system administrator may use one or more utility programs to determine the state of disk configuration and/or formats so that the disk information may be saved. Additional programs and techniques may be used to record a list of operating system files, data, and other software installed on the system. The administrator may also record the types of various devices and settings thereof installed in a system. Backing up a system's state is thus a formidable task.

Similarly, the process of restoring a system involves the use of this recorded information, along with an operating system setup program, thus making restoration a complicated process. Moreover, if the original system is replaced with non-identical hardware, (e.g., a larger disk, a new CD-ROM, Hard Disk Controller, and/or Video Card) then additional complications may arise because much of the saved state information may no longer apply to the new system configuration. For example, if a system fails and the data and files are restored to a non-identical system, many hours may have to be spent adjusting and configuring the system to work, using a variety of different programs and utilities. In sum, present system recovery (backup and restore) involves proprietary and custom crafted solutions that are not common and extensible. Instead, providers of backup and restore programs each redefine an environment, process, and syntax to enable the recovery of the system.

As a result, whenever a failure makes a system non-bootable, the process to reconstruct the system's previous state is error prone and lengthy. This can cause serious problems, particularly with computer systems used in critical roles (such as a file server) wherein the time required to get the computer system operational after a failure is very important.

SUMMARY OF THE INVENTION

Briefly, the present invention provides a system recovery framework / architecture for backup and restore, thereby providing an extensibility mechanism, such as for third party vendors to integrate with their programs. The framework defines a common process, environment, and syntax for straightforward integration into system recovery programs. Backup programs integrate with this framework by collecting and writing appropriate information to be used during system recovery in the proper format. The information can be collected using available APIs including Backup APIs, Query Hard Disk State APIs, NT Registry APIs and a Query System Catalog State mechanism that finds and reads the description of the set of files that cannot be modified.

Saved file data may be stored in whatever format the backup program needs and requires, while the state information and additional files are written out in defined and documented syntax and format, i.e., to a System Information File (sif file). This file is a text file that specifies the hard disk state of the system and the location, size, and type of key partitions where the Operating System (Windows) is located. The sif file also includes instructions for specifying what programs are to be launched during the restore phase, and any commands that need to be run in error handling cases. The sif file also includes (or references) any additional drivers or files to copy to assist in the restore process, such as a custom driver. System catalog information is also backed up, and stores a catalog that contains the enumeration of the required system files that need to be present in the system and that cannot be altered.

A framework for restoring is also provided, and includes reading the sif file to restore the disk partition state, creating a common environment, configuring (initializing) the common environment, including detecting any new devices of the system, and handling any differences. When the environment is configured, the programs specified in the sif file are then run to restore the remainder of the system.

Other advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram representing a computer system into which the present invention may be incorporated;

FIG. 2 is a block diagram representing exemplary components for backing up system state and other system information in accordance with an aspect of the present invention;

FIG. 3 is a block diagram representing exemplary components for restoring system state and other system information in accordance with an aspect of the present invention;

FIGS. 4A - 4C comprise a representation of a file for storing system state information in accordance with an aspect of the present invention;

FIG. 5 is a flow diagram generally representing an overall process for backing up and restoring a system in accordance with an aspect of the present invention;

FIG. 6 is a flow diagram generally representing a process for backing up system state and other system information in accordance with an aspect of the present invention;

FIG. 7 is a flow diagram generally representing a process for restoring system state information in a first phase in accordance with an aspect of the present invention; and

FIG. 8 is a flow diagram generally representing a process for restoring system state information and other system data in a second phase in accordance with an aspect of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

EXEMPLARY OPERATING ENVIRONMENT

FIGURE 1 and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types.

Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20 or the like, including a processing unit 21, a system memory 22, and a system bus 23 that couples various system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS), containing the basic routines that help to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. The personal computer 20 may further include a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD-ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs) and the like may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35 (preferably Windows 2000), one or more application programs 36, other program modules 37 and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port or universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor 47, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49. The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local network 51 through a network interface or adapter 53. When used in a WAN networking environment, the personal computer 20 typically includes a modem 54 or other means for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

BACKUP OF SYSTEM STATE

The present invention a system recovery framework for backup and restore that provides an extensibility mechanism, such as for third-party vendors to integrate into backup and restore programs. This framework defines a common process, environment, and syntax that allow vendors to easily integrate their programs into the system recovery process. As will be understood, authors of backup and restore applications may thus focus on the backup and restore features of their product, rather than dealing with creating environments or processes to run their programs. Also described is Automated System Recovery (ASR), an integrated mechanism for the backup and restoration of system state. The ASR mechanism provides a single coherent and structured mechanism for backing up and restoring system state. The ASR mechanism is further described in the copending United States Patent Application entitled, "*Automated System Recovery via Backup and Restoration of System State,*" assigned to the assignee of the present invention, filed concurrently herewith, and hereby incorporated by reference in its entirety.

Turning to FIG. 2 of the drawings, there is shown a backup component including a backup program (process) 60 for providing an integrated mechanism for determining and storing state information 62 in accordance with one aspect of the present invention. As described below, the state information 62 is later used for restoring a system (e.g., the system 20) after a failure in which the system 20 cannot reboot. Note that the restored system may be an entirely new system, the same system with the same components, (e.g., the boot volume or system volume inadvertently changed, preventing reboot), or essentially the same system but with one or more new components, (e.g., a new hard disk controller was substituted). In any event, the state is saved such that the failed system may be restored to an operational system, and thus the present invention comprises two primary components, one for backup, and one for restore.

In accordance with one aspect of the present invention and as shown in FIG. 2, in addition to backing up the file data 64 normally backed up, e.g., the application programs 66 (FIG. 3) and/or data files 67, the backup program / process 60 of the present invention records the system state 62. To this end, the backup component (e.g., via the backup program 60) copies and stores the state that intrinsically defines the computer system 20 for potential and future recovery, in various areas. Note that the backup program 60 is ordinarily executed on a regular (maintenance) basis by an administrator or the like, and also should be run whenever the system configuration changes, e.g., a new hard disk is added.

In keeping with the present invention, a first area for backing up includes the system state, which is an underlying description of the system, separate from the actual operating system and data files, i.e., the information that intrinsically defines the configuration of a computer system. The system state includes the hard disk configuration (i.e., disk structure and layout) such as the number of hard disks on the system, number of disk partitions on the system and partition format types (such as NTFS, FAT, or FAT32). The hard disk configuration also includes descriptions of where the partitions start and stop on the disk (partition offsets), so that the disk or disks can be recreated exactly during the restore phase. Also stored is the location of the operating system partition or partitions required to start a machine, e.g., where the Windows® 2000 boot volume and system volume are installed. These partitions are identified so the proper associations with them can be restored. Also, in the event that the partitions and disks are different during the restore phase, these boot and system partitions will be restored as first priority. Of course, if some storage mechanism other than a hard disk is used, equivalent data will be maintained therefor.

As will be described below, the system state 62 is recorded on a medium (e.g., a floppy disk 68) that is readily accessible to (i.e., readable by) a newly operational, but not yet restored machine. As used herein, the readily-readable medium will be described as a floppy disk 68, but as can be readily appreciated, the medium may alternatively comprise a read-writeable CD-ROM, network storage (e.g., in a directory of files), flash memory card, wired or wireless telephone connection, smart card and virtually anything else capable of recording and/or transmitting information to a computer system for use by a restore process 70 (FIG. 3). Note that the computer system need not be fully configured for operation at that time, e.g., the hard disks thereof need not be formatted, nor all the various device drivers loaded when the medium 68 is access for restore. Part of the system state, e.g., certain selected, less-essential operating system files, may also be stored to a secondary backup device 74 (e.g., a tape drive or hard disk, such as a network disk drive) that generally stores relatively larger amounts of information. Similarly, as described hereinafter, the backup device 74 will be primarily described as a tape drive, however any medium capable of storing the appropriate amount of information may be used, including a hard disk drive or even the same readily-accessible medium 68 that stores the primary system state information. For example, a single two-gigabyte optical disk may store all of the necessary state information and data for a low-end system.

Thus, a second area to be backed up includes the operating system 72 and data files 64, e.g., the files on the computer and their associated properties including a complete set of the operating system files. As described above, this generally larger amount of data is ordinarily written to the backup device 74, e.g., a tape drive.

A third area to be backed up includes a description of the operating system and data files, e.g., the files on the computer and their associated properties, including a complete set of the recovery information, and a list of what programs should be executed during the restore phases along with other information needed for the restore. More particularly, backing up this area includes specifying the programs to copy and execute on restore, any error handling, and any special driver files to load. For example, if the system has devices that require special drivers or support, information about these drivers can be stored on the floppy disk 68 so that these drivers will be loaded during the restore phase. Thus, if the backup device 74 has a special driver, that driver is identified in this area. The actual driver code file may be recorded on the floppy disk 68, the backup device 74, or on some combination of both, however as can be appreciated, if a special driver is needed for accessing the backup device 74, it will be written to the floppy disk 68 rather than to the backup device 74.

To collect the information, the backup program 60 calls a number of well-documented APIs (application programming interfaces), and performs other operations as described below. As a result, any backup program can integrate with the present invention framework as long as the backup program collects and writes the required information to be used during system recovery.

A first set of information is collected via a Backup() API 80 that provides a set of common and well-documented functionality to collect information and state on a running system. This API 80, the Backup Read() API (and similarly the Backup Write() API) handles some of the collection of information on programs and processes 82 that are in use and whose state is changing.

A second operation performed by the backup program 60 is to query the system for the hard disk state, i.e., the disk geometry. To this end, common, documented APIs 84 exist (publicly available in the Software Development Kit and/or Device Driver Kit from Microsoft® Corporation, Redmond, Washington) and are provided to determine the number of partitions and volumes on one or more hard disks 86 connected to the system. These APIs 84 also provide information on partition types, formats, labels, and offsets, e.g., where the partitions are located on the disk, where the boot volume and system volume are located and so on. For example, a partition table may be constructed that identifies the number of sectors, which sectors are in which partition, which and where various volume managers are used, and so on. As described below, this information is saved with the state files 62 in a special "sif" (System Information File) format.

A third operation performed by the backup program 60 is to collect system registry 88 information, again via common (NT Registry) APIs 90. As is well-known, the system registry 88 essentially comprises a database of properties, settings and other information used by the operating system and applications at various times to perform various functions. For example, the registry 88 includes information on specific devices and drivers installed on this system, such as the hard disk controller and audio card. Via this set of common APIs 90 and conventions for retrieving the data stored in the registry 88, the saved registry may be recreated and/or adjusted on a restored system.

A fourth operation that the backup program 60 performs is to query the system catalog state via a query component 92. In general, this query component 92 finds and reads the description of the set of files 94 that cannot be modified by any application or system service, and have to be identical to those present in the operating system's CD since its release. Such files may be marked with a file attribute or the like for identification, in accordance with system file protection (SFP) features.

It should be noted that other information may be collected in a similar manner, thereby extending the backup and restore capabilities. For example, one or more functions, such as performed via an API, may obtain network state information, whereby the precise network state may be reconstructed on a restored machine. Video cards, RAM, and other devices on may have state saved therefor in a similar manner, whereby if the a new system has at least the same sizes as the original machine, the original machine may be reconfigured. Also, some programs (such as IIS) have their own database of information. APIs may similarly back up this information on a running system.

In accordance with another aspect of the present invention, once this information is collected, it is written out to the floppy disk 68 and/or backup device 74 in defined formats to be used by the restore component (FIG 3). The saved information collected from the Backup APIs 80, and anything else that needs to be restored, e.g., application programs 66 and/or their data 67 (FIG. 3), is written in whatever format the backup / restore programs need and require. The system configuration portion of the state information is written out to in a defined "sif" format, along with additional files, as described below. The catalog information 96 is also written to the floppy disk drive 68.

A preferred format for a sif file 98 written by the backup application 60 is a text file that matches a particular syntax and format. This file comprises a self-contained summary of the system state that specifies, among other things, the hard disk state of the system and the location of key partitions where the operating system (e.g., Windows® 2000) is located. A sif file 98 (FIGS. 4A - 4C) also includes instructions specifying which program or programs are to be launched during the restore process, and any commands that need to be run for error handling. The sif file 98 (Dr_state.sif) also identifies any additional drivers or files to copy to assist in the restore process, e.g., a custom driver.

FIGS. 4A - 4C represent a sample sif file 98 written out for an exemplary system. Note that FIGS. 4A - 4C represent the same sif file 98, however the file spans three drawing figures to assist in readability. Further, note that in FIGS. 4A - 4C, the "~~~" characters in the text indicate where part of a data string was omitted to better fit the text horizontally, and also that some of the strings as shown occupy two lines even though there is no hard return at the end of the first of those lines. In any event, as shown in FIGS. 4A - 4C, the sif file 98 comprises a data structure including fields of data, arranged as version information, commands, disk information, partition information, volume information and so on. Of course, the sif data structure may be divided into a plurality of files, e.g., one file for the disk information and another for the commands, and so on.

When later interpreted by a restore process 70 (FIG. 3), the sif file 98 will be used to reconstruct the state of the machine prior to failure. Note that the sif file 98 stores information such as the size of the disk, whether the volume was mirrored, striped, the volume managers used and other pieces of information that enable the disk state to be rebuilt on another system. Also, some manufacturers reserve a partition on the disk which may need to be read sector by sector to preserve it for later preservation, and although this data is ordinarily not stored in the sif file 98, the sif file 98 may include the information as to how to restore it.

Also written out by the backup program is the catalog 96 that comprises an enumeration of the required system files that need to be present in the system, and that cannot be altered. This ensures that the appropriate files will be available for the new system configuration. When the above tasks are finished, the system information has been recorded to one or more suitable backup devices. At this time, the recorded information can be used to recover a system.

RESTORATION OF SYSTEM STATE

As described above, a second, restore component of the present invention is directed to restoring the system state after a failure. This second, restore component of the Automated System Recovery (ASR) process includes two phases, whereby each phase ensures that the right data and configuration is restored properly and in the correct order.

The ASR framework / mechanism provides a simple and a fast solution to the problem of restoring a system after failure by executing a set of documented and defined operations to restore the system. The information provided by the backup program (or programs) 60 will be used to restore the system.

The extensible framework for restore is generally shown in FIG. 3, wherein a restore process 70 is run in the event of failure to restore a system from recorded backup information. As generally represented in FIG. 3, a first phase of the restore process includes running the operating system setup CD 102 to start Automated System Recovery (ASR) 104. Running the operating system CD 102 loads the necessary drivers and information to view and access critical parts of the computer such as the hard disks. Of course, some mechanism (i.e., the system ROM) also enables the CD-ROM, and/or floppy disk drive to be initially accessed so that the setup CD 102 may be initially run. The use of a setup CD 102 to install an operating system on an operational machine is well known, and thus will not be described in detail hereinafter except to mention that ASR may be integrated with the operating system startup CD 102 in a straightforward manner. This integration provides a standard point for starting the restoration of the system.

During the setup process, the system operator is given an opportunity to select and run ASR 104. When selected, ASR 104 prompts the operator for the floppy disk 68 (or other medium) that includes the information saved during the backup phase. The sif file 98 thus may guide various aspects of the setup process.

In a first phase of recovery, ASR runs the restore process 70 to read the sif file 98 in order to compare and restore the underlying disk state of the system, with respect to disk partitions, layout, and offsets as specified in the sif file 98. For example, the restore process 70 scans the disk partition or partitions and volume or volumes, using the information saved in the sif file 98 to compare the current state of the disk partitions and volumes on the new system with the former system. If there are any differences, the disk and volume state are restored according to the saved information, e.g., the disk 86_r (where subscript "r" indicates the restored counterpart to the former system) is appropriately partitioned and the appropriate volumes set up. The partitioning of disks and creation of volumes is well known, and is thus not described in detail herein for purposes of simplicity. Note that if the disks and the volumes existing on the system are not identical to the original disks and volumes that were present when the backup was made, the volume and disk information is adjusted and restored to the best possible extent. For example, if the former system had two partitions, each on separate four-gigabyte disks, equivalent partitions may be established on a portion of a fourteen-gigabyte disk of a newly configured system. The adjusting for such differences is further described in the copending United States Patent Application entitled, "*System Recovery By Restoring Hardware State on Non-Identical Systems*," assigned to the assignee of the present invention, filed concurrently herewith, and hereby incorporated by reference in its entirety.

Once the underlying system state is restored, an environment is created so that the operating system files and data files 68 described above may be restored. To this end, a restore environment is created by copying the files that are required to run the programs that will restore the remainder of the files. More particularly, the set of files required to start a graphical user interface (GUI) environment that will let the restore program or programs 100 (written for the Windows 32-bit platform run and execute) are copied to the boot volume 106 and system volume 108, i.e., at least the core operating system files are copied. Note that these files may be on the same volume. Also, the restore application program 100 is copied to the hard disk (unless already present on an intact volume). The registry 88_r may also be copied at this time. Once these files are copied, the system is restarted.

After the files are copied and the reboot occurs, the next phase begins by initializing the environment. Initializing involves re-detecting any new devices that have been attached to the system, and installing any needed drivers 110 or files used by the restore program or programs 110. The automatic detection of hardware and/or installation of corresponding software by an operating system is well known, (e.g., plug and play technology) and is thus not described herein in detail. Note that at this time, the system catalog 96 has also been accessed, and is used to prevent the restoration of a different version of any of the files that are identified in the catalog 96.

To further configure the environment for the programs, ASR (via the operating system) thus detects and installs drivers 110 and support for devices installed on the system. This ensures that devices (such as a tape drive) are available for the restore program 100. These drivers and support for these drivers are normally loaded from the operating system CD, however as described above, any specialized drivers and support therefor that may have been present were saved as part of the backup process to ensure their availability, and thus may be loaded from the floppy disk 68 or the like. Any changed information is recorded in the registry 88_r.

After the environment is configured, the restore program or programs 110 are run according to the instructions that were saved in the sif file 98 during the backup process. This will usually result in restoring the operating system and data files from the backup device 74. In the event of an error condition, the instructions specified will be executed. Note that during restoration, if a volume is found intact, that volume is not restored, potentially saving considerable time. For example, it is possible that only the boot volume was damaged, whereby only the boot volume 106 need be restored to restore the system. In any event, after restoring the data files, the system 20_r is then restarted, after which the restoration and recovery is complete.

The present invention will now be summarized with respect to the flow diagrams of FIGS. 5 - 8. First, as represented at step 500 of FIG. 5, the system is backed up during regular maintenance, or whenever a configuration change is made, for example, if new hardware is installed. To this end, as shown in FIG. 6, the Backup() APIs 80 are called (step 500), the hard disk state is queried (step 602), the registry APIs 90 are called (step 604) to read the system registry 88, and the system catalog state is queried to catalog those files that are not allowed to change.

Step 608 represents the writing out of the file and other data (collected at step 600) to be backed up by the backup program 60 according to its desired format. In keeping with the present invention, at step 610, the state data collected via steps 602 - 604 is written out to the floppy disk 68, including the sif file 98, the list of programs and commands needed for restoration, and any custom drivers. Lastly, step 612 represents the writing out of the catalog information 96 gathered by step 606. Note that these steps need not be performed in the order shown in FIG. 6. For example, the various pieces of information may be recorded as soon as each piece is collected, rather than first collecting the information before performing any writing.

Returning to FIG. 5, sometime after a backup at step 500, the system may fail (step 502) in a manner in which it cannot be rebooted. At this time, restoration via the backed up state information 62 is desired. Whether restoring to an entirely new system, a repaired system or the same system, the restore process 70 begins its first phase at step 504 when the operator runs an appropriate setup CD 102 (step 700). During execution of the setup, when prompted, the operator selects automated system recovery 104, and inserts the floppy disk 68 (or otherwise provides the system with the backup medium) at step 702.

As represented by step 704, the ASR mechanism (restore process 70) of the present invention next scans the disk partitions, volumes and so forth of the current system to compare with the state information stored for the previous system in the sif file 98 on the floppy disk 68. As described above, any system differences are detected at step 706, and adjusted for at step 708.

At step 710, the disk partition state is restored, for example, to establish the boot volume 106 and/or system volume 108 at their appropriate locations on the disk 86_r. Then, as described above, the files needed for providing a restoration environment (e.g., to run Win32 applications) are copied from the floppy disk 68 to the hard disk at step 712.

Returning to FIG. 5, at step 506, the system is rebooted, after which the system is ready for the second phase of restoration, as generally represented via step 508 and in the steps of FIG. 8. At step 800 of FIG. 8, the restoration environment is configured by installing device drivers 110, device support and so on, particularly for any devices needed by the restore program 100. For example, at this time, a device driver for the tape drive on which the rest of the backed up data is installed.

Step 802 represents the running of the restore application program 100 that is the counterpart of the program used to back up the operating system files 72, applications 66 and data 67. If any errors are detected at step 804, step 806 is executed to handle the error, including executing any special error handling instructions that were saved during backup. Lastly, when restoration is complete, the system is restarted to clean up and remove the restoration environment, as represented by step 808. When restarted, the restoration and recovery is complete. Note that in addition to recovery, the present invention also may be used for replication of systems.

As can be seen from the foregoing detailed description, there is provided a method and framework that enables the backup and restoration of a failed system in an automated and efficient manner. In accordance with the framework, a backup component copies and stores the state that intrinsically defines the configuration of the computer system for potential and future recovery, and a two-phase restoration process uses the backed-up information to restore the system.

While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form or forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

WHAT IS CLAIMED IS:

1. A method for recovering from a failure of a computer system, comprising:

collecting state information of the system, the state information including hard disk state, registry state and at least one instruction for launching at least one restore program;

writing the state information to a medium in a defined format;

reading the state information from the medium and providing a restoration environment therefrom; and

launching the at least one restore program.

2. The method of claim 1 wherein collecting state information includes calling at least one API.

3. The method of claim 1 wherein collecting state information includes obtaining registry information via at least one API call.

4. The method of claim 1 wherein collecting state information includes obtaining hard disk geometry data via at least one API call.

5. The method of claim 1 wherein the defined format includes data stored as text.

6. The method of claim 1 wherein the defined format includes state information arranged into fields.

7. The method of claim 6 wherein one of the fields includes the instructions for launching the at least one restore program.

8. The method of claim 6 wherein one of the fields includes data identifying hard disks of the backed up system.

9. The method of claim 8 wherein one of the fields includes partitioning data of the hard disks.

10. The method of claim 8 wherein one of the fields includes volume data of the hard disks.

11. The method of claim 8 wherein one of the fields includes offset data of the hard disks.

12. The method of claim 1 wherein writing the state information includes saving information to a floppy disk.

13. The method of claim 1 wherein writing the state information includes saving information to a CD-ROM.

14. The method of claim 1 wherein writing the state information includes saving information to at least one network file.

15. The method of claim 1 wherein reading the state information includes receiving the state information over a transmission medium.

16. The method of claim 1 wherein providing a restoration environment includes copying operating system files to a hard disk.

17. The method of claim 1 wherein providing a restoration environment includes installing a device driver.

18. The method of claim 1 further comprising, determining a set of unchangeable files, and writing a catalog of that set to the medium.

19. A computer-readable medium having computer executable instructions for performing the method of claim 1.

20. A computer-readable medium having stored thereon a data structure, comprising:

a first set of data representing storage devices;

a second set of data representing partition information of the storage devices represented in the first set of data;

a third set of data representing volume information of the storage devices represented in the first set of data; and

a fourth set of data representing at least one command for restoring data to at least one storage device configured based on the information in the first, second and third sets of data.

21. The computer-readable medium having stored thereon the data structure of claim 20, wherein each set of data is separated into a distinct field from one another.

22. The computer-readable medium having stored thereon the data structure of claim 20, further comprising, a fifth set of data having version information therein.

23. The computer-readable medium having stored thereon the data structure of claim 20, further comprising, a fifth set of data having volume state information therein.

24. The computer-readable medium having stored thereon the data structure of claim 20, further comprising, a fifth set of data having removable media information therein.

25. A method for backing up a computer system for subsequent restoration, comprising:

collecting restore information, including hard disk configuration information, registry information, a catalog of unchangeable files and at least one instruction for launching at least one restore program; and

recording the restore information to at least one medium.

26. The method of claim 25 wherein collecting the restore information includes calling at least one API.

27. The method of claim 25 wherein the hard disk configuration information includes partition and volume information.

28. The method of claim 25 wherein the hard disk configuration information includes information indicative of where key operating system files are located.

29. The method of claim 28 wherein the information indicative of where key operating system files are located includes the location of a boot volume.

30. The method of claim 28 wherein the information indicative of where key operating system files are located includes the location of a system volume.

31. The method of claim 25 wherein the registry information includes data corresponding to hardware installed in the system.

32. The method of claim 31 wherein the data corresponding to hardware installed in the system includes device driver information.

33. A computer-readable medium having computer executable instructions for performing the method of claim 25.

34. A method for restoring a computer system, comprising:

reading restore information, the restore information including saved hard disk configuration information, registry information, a catalog of unchangeable files and at least one instruction for launching at least one restore program; and

restoring the computer system by configuring at least part of the system based on the restore information, and by executing the at least one instruction.

35. The method of claim 34 wherein configuring at least part of the system includes scanning for actual hard disk information of the computer system and comparing the actual hard disk information to the saved hard disk configuration information.

36. The method of claim 34 wherein the saved hard disk configuration information includes partition and volume information, and wherein configuring at least part of the system based on the restore information includes reconfiguring at least one hard disk of the computer system to correspond to the partition and volume information.

37. The method of claim 34 wherein the hard disk configuration information includes information indicative of where key operating system files are located, and wherein configuring at least part of the system based on the restore information includes copying operating system files to a hard disk location based on the information.

38. The method of claim 34 wherein configuring at least part of the system based on the restore information includes creating a boot volume.

39. The method of claim 34 wherein configuring at least part of the system based on the restore information includes creating a system volume.

40. The method of claim 34 wherein configuring at least part of the system based on the restore information includes updating a registry.

41. The method of claim 34 wherein configuring at least part of the system based on the restore information includes installing a device driver.

42. The method of claim 34 wherein executing the at least one instruction includes launching a restore program.

43. A computer-readable medium having computer executable instructions for performing the method of claim 34.

44. A system for recovering from a failure of a computer device, comprising:

a medium;

a backup process for collecting restore information of a first system, the restore information including hard disk configuration information and at least one instruction for launching a restore program, the backup process writing the restore information to a medium in a defined format; and

a restore process for reading the state information from the medium, configuring a second system based on the state information, and launching the restore program.

45. The system of claim 44 further comprising a backup device, and wherein the backup process writes information for enabling the backup device to the medium.

46. The system of claim 45 wherein the information for enabling the backup device includes a device driver.

47. The system of claim 45 wherein the backup process stores at least some information of the first system on another medium via the backup device.

48. The system of claim 47 wherein the restore process restores at least some information to the second system from the other medium.

ABSTRACT

A system recovery method and framework for backing up and restoring a system that cannot reboot. The framework defines a common process, environment, and syntax, whereby backup programs integrate with this framework by collecting and writing appropriate information to be used during system recovery in the proper format. The format is a System Information File, a text file that specifies the hard disk state of the system and the location of key partitions where key components of the operating system are located. The file also includes instructions for specifying programs to launch during the restore phase, and any commands that need to be run in error handling cases. The file also includes or references any additional drivers or files to copy to assist in the restore process. A framework for restoring is also provided, and includes reading the file to restore the disk partition state, creating a common environment, and configuring (initializing) the common environment. When the environment is configured, the programs specified in the file are run to restore the remainder of the system.

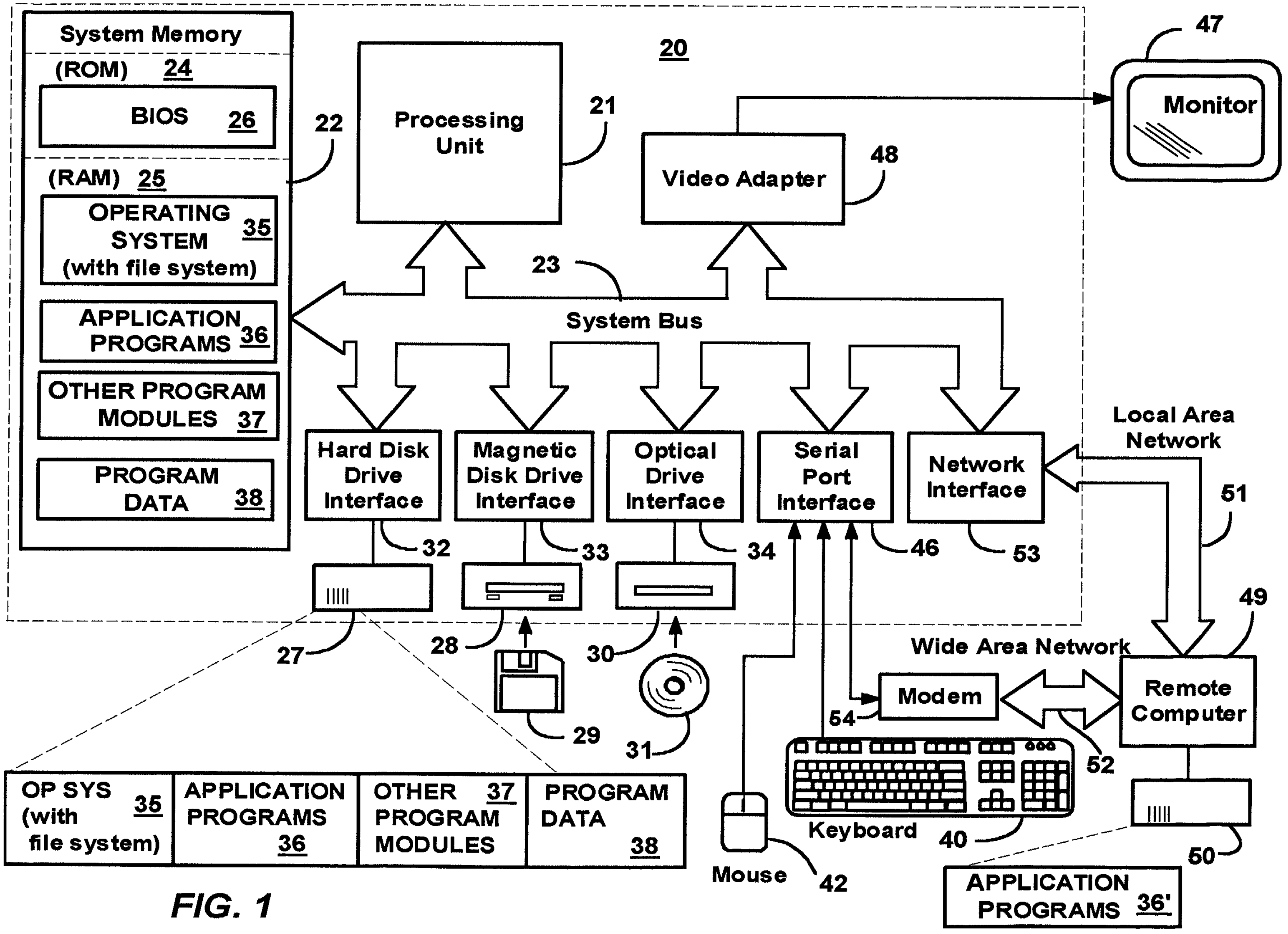

System Memory
(ROM) 24
BIOS 26
(RAM) 25
OPERATING SYSTEM (with file system) 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38
22
Processing Unit
21
20
Video Adapter
48
Monitor
47
23
System Bus
Hard Disk Drive Interface
32
Magnetic Disk Drive Interface
33
Optical Drive Interface
34
Serial Port Interface
46
Network Interface
53
Local Area Network
51
49
27
28
29
30
31
Wide Area Network
Modem
54
Remote Computer
52
Keyboard
40
Mouse
42
50
APPLICATION PROGRAMS 36'
OP SYS (with file system) 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38

FIG. 1

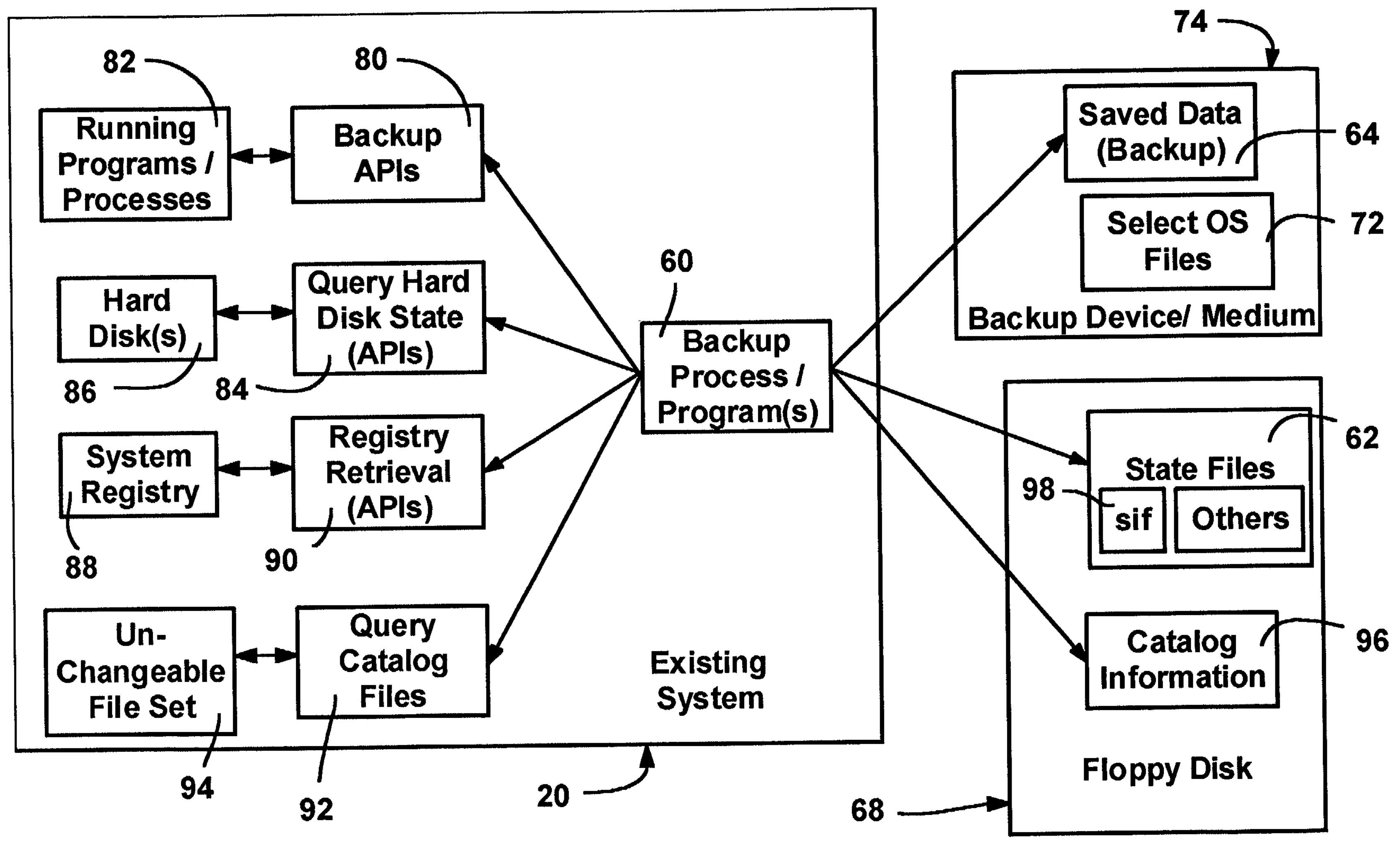
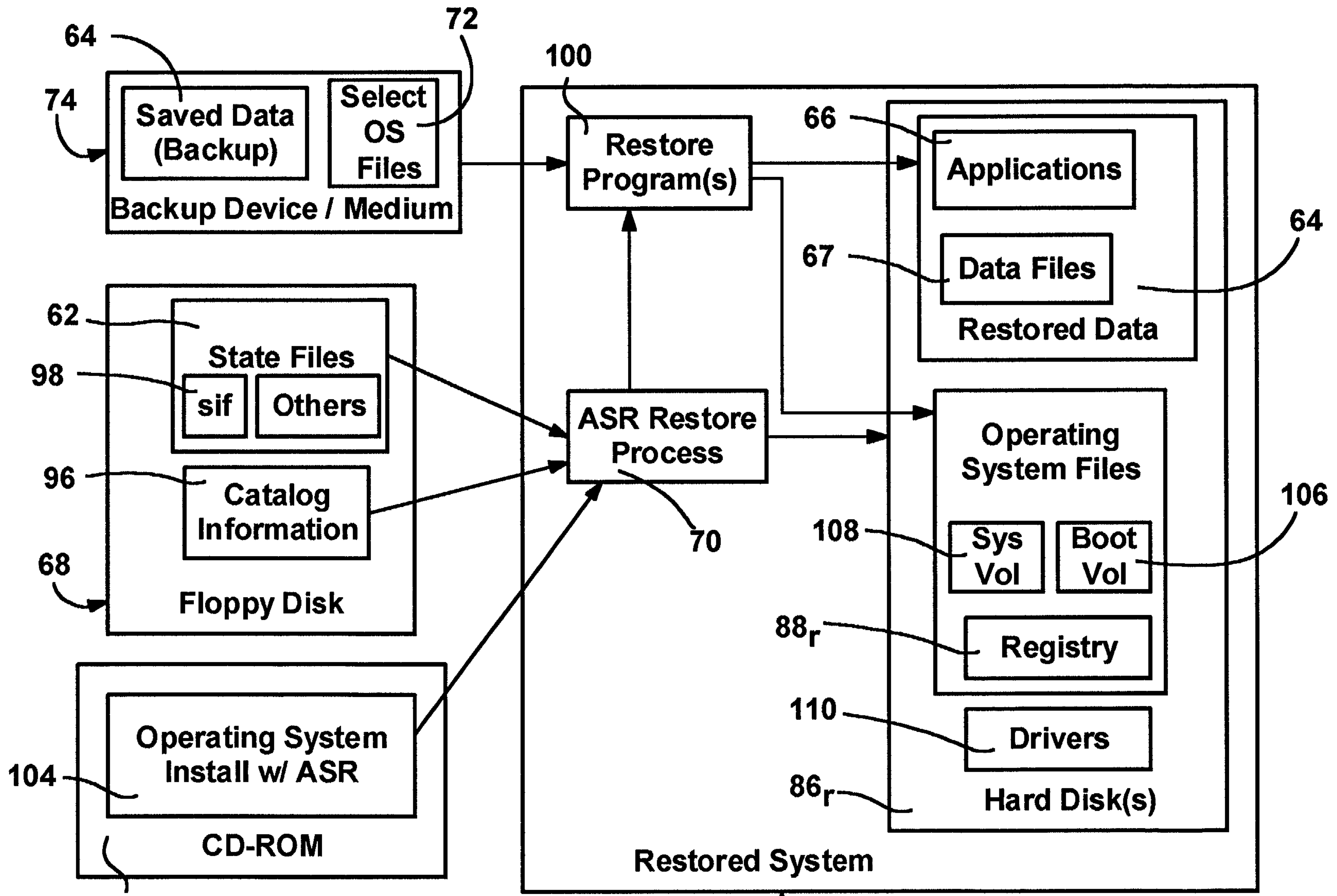
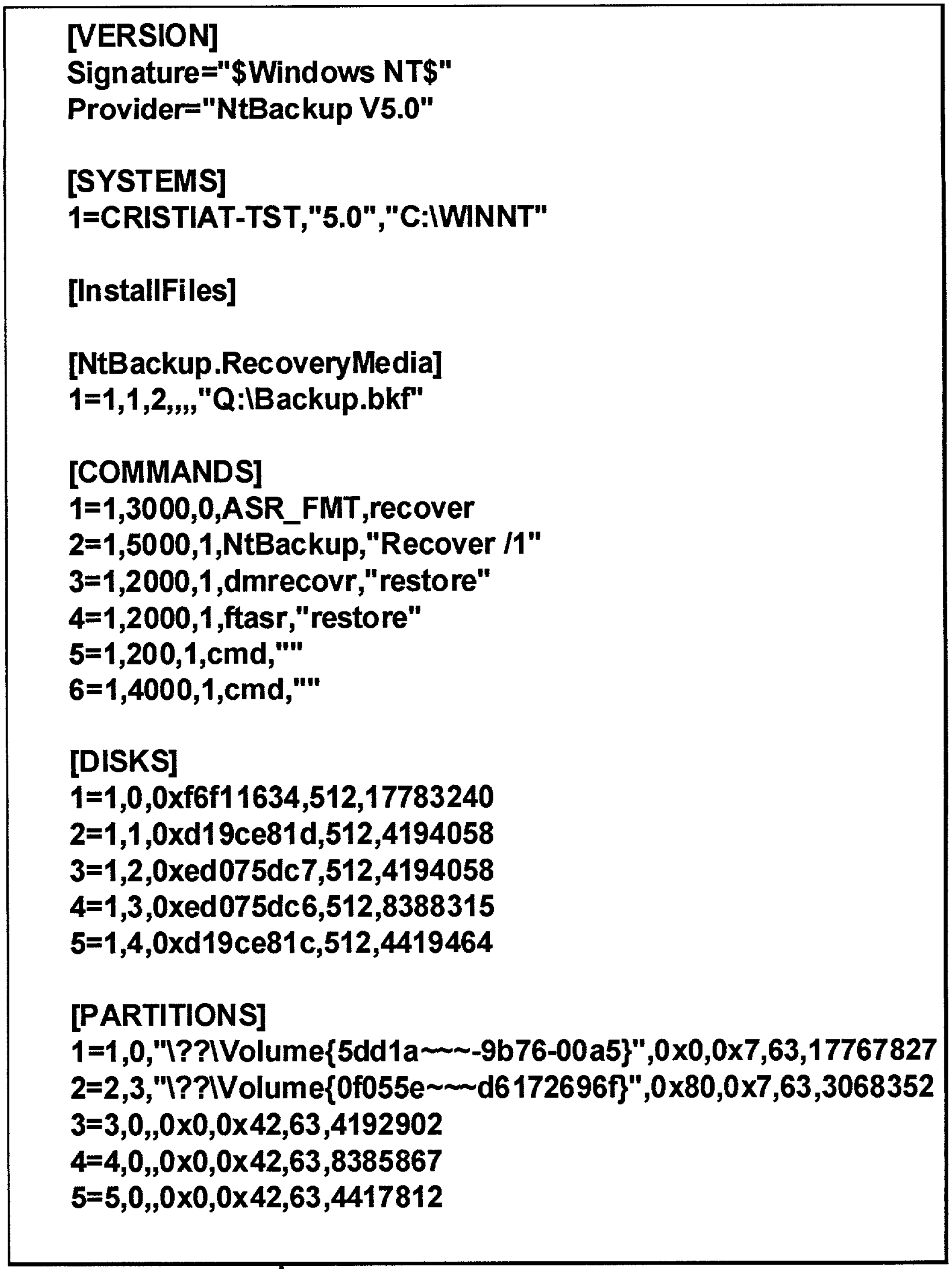

Running Programs / Processes
82
Backup APIs
80
Hard Disk(s)
86
Query Hard Disk State (APIs)
84
System Registry
88
Registry Retrieval (APIs)
90
Un-Changeable File Set
94
Query Catalog Files
92
Backup Process / Program(s)
60
Existing System
20
Saved Data (Backup)
64
Select OS Files
72
Backup Device/ Medium
74
State Files
62
sif
98
Others
Catalog Information
96
Floppy Disk
68

FIG. 2



64
72
74
Saved Data (Backup)
Select OS Files
Backup Device / Medium
100
Restore Program(s)
66
Applications
64
67
Data Files
Restored Data
62
State Files
98
sif
Others
96
Catalog Information
68
Floppy Disk
ASR Restore Process
70
Operating System Files
106
108
Sys Vol
Boot Vol
88_r
Registry
110
Drivers
86_r
Hard Disk(s)
104
Operating System Install w/ ASR
CD-ROM
102
Restored System
20_r

FIG. 3



```
[VERSION]
Signature="$Windows NT$"
Provider="NtBackup V5.0"

[SYSTEMS]
1=CRISTIAT-TST,"5.0","C:\WINNT"

[InstallFiles]

[NtBackup.RecoveryMedia]
1=1,1,2,,,,"Q:\Backup.bkf"

[COMMANDS]
1=1,3000,0,ASR_FMT,recover
2=1,5000,1,NtBackup,"Recover /1"
3=1,2000,1,dmrecovr,"restore"
4=1,2000,1,ftasr,"restore"
5=1,200,1,cmd,""
6=1,4000,1,cmd,""

[DISKS]
1=1,0,0xf6f11634,512,17783240
2=1,1,0xd19ce81d,512,4194058
3=1,2,0xed075dc7,512,4194058
4=1,3,0xed075dc6,512,8388315
5=1,4,0xd19ce81c,512,4419464

[PARTITIONS]
1=1,0,"\??\Volume{5dd1a~~~-9b76-00a5}",0x0,0x7,63,17767827
2=2,3,"\??\Volume{0f055e~~~d6172696f}",0x80,0x7,63,3068352
3=3,0,,0x0,0x42,63,4192902
4=4,0,,0x0,0x42,63,8385867
5=5,0,,0x0,0x42,63,4417812
```

(cont.)

FIG. 4A

(cont.)

```
[VOLUMES]
1=1,"\??\Volume{0f055eb4-eb6e-11d2-b895-806d6172696f}",
"\Dos Devices\C:",NTFS,,0x800
2=1,"\??\Volume{5dd1a967-ebb4-11d2-9b76-00a0c9063765}",
"\Dos Devices\Q:",NTFS,Backup,0x1000
3=1,"\??\Volume{5dd1a969-ebb4-11d2-9b76-00a0c9063765}",
"\Dos Devices\V:",NTFS,Spanned,0x400
4=1,"\??\Volume{5dd1a96b-ebb4-11d2-9b76-00a0c9063765}",
"\Dos Devices\S:",NTFS,Stripe,0x400
5=1,"\??\Volume{5dd1a96d-ebb4-11d2-9b76-00a0c9063765}",
"\Dos Devices\P:",NTFS,RAID-5,0x400
6=1,"\??\Volume{5dd1a96f-ebb4-11d2-9b76-00a0c9063765}",
"\Dos Devices\M:",NTFS,Mirror,0x400

[RemovableDisks]
1=1,"\Device\CdRom0","\??\Volume{31691342-eb70-11d2-
      9b75-806d6172696f}","\DosDevices\D:"
2=1,"\Device\Floppy0","\??\Volume{31691343-eb70-11d2-
      9b75-806d6172696f}","\DosDevices\A:"

[LDM.VolumeState]
  1852, 59,"LDM
Configuration","Microsoft1","31691341-eb70-11d2-9b75
    -806d6172696f"
"Group","e79bc5a0-ebb4-11d2-9b76-00a0c9063765","
Cristiat-tstDg0","primary","0.1095"
"Disk","Disk1","Active","-318284345","e79bc5a1-~~~5&0&030"
"Disk","Disk2","Active","-318284346","e79bc5a2-~~~5&0&040"
"Disk","Disk3","Active","-778246116","e79bc5a3-~~~5&0&050"
"Volume","2a0b4281-~~~,"Mirror","184~~~b76-00a0c9063765}"
"Plex","Volume1-01","Span"
```

(cont.)

98

FIG. 4B

(cont.)

```
"Subdisk","Disk1-01","819200","0","0","Disk1","0"
"Subdisk","Disk2-01","614400","0","819200","Disk2","0"
"Subdisk","Disk3-01","409600","0","1433600","Disk3","0"
"Volume","61ab66d0-ebb5-11d2-9b76-00a0c9063765","Mirror",
"1843200","","0x7","","{5dd1a96b-ebb4-1~~~6-00a0c9063765}"
"Plex","Stripe1-01","Stripe","3","128"
"Subdisk","Disk1-02","614400","0","0","Disk1","819200"
"Subdisk","Disk2-02","614400","1","0","Disk2","614400"
"Subdisk","Disk3-02","614400","2","0","Disk3","409600"
"Volume","92b61350-ebb5-11d2-9b76-00a0c9063765","Raid-5",
"1638400","","0x7","","{5dd1a96d-ebb4-1~~~6-00a0c9063765}"
"Plex","Raid1-01","Raid-5","3","128"
"Subdisk","Disk1-03","819200","0","0","Disk1","1433600"
"Subdisk","Disk2-03","819200","1","0","Disk2","1228800"
"Subdisk","Disk3-03","819200","2","0","Disk3","1024000"
"Volume","061c5f40-ebb6-11d2-9b76-00a0c9063765","Mirror",
"1228800","","0x7","","{5dd1a96f-ebb4-1~~~6-00a0c9063765}"
"Plex","Volume2-01","Span"
"Subdisk","Disk1-04","1228800","0","0","Disk1","2252800"
"Plex","Volume2-02","Span"
"Subdisk","Disk2-04","1228800","0","0","Disk2","2048000"


[FT.VolumeState]
0
```

98

FIG. 4C

FIG. 5

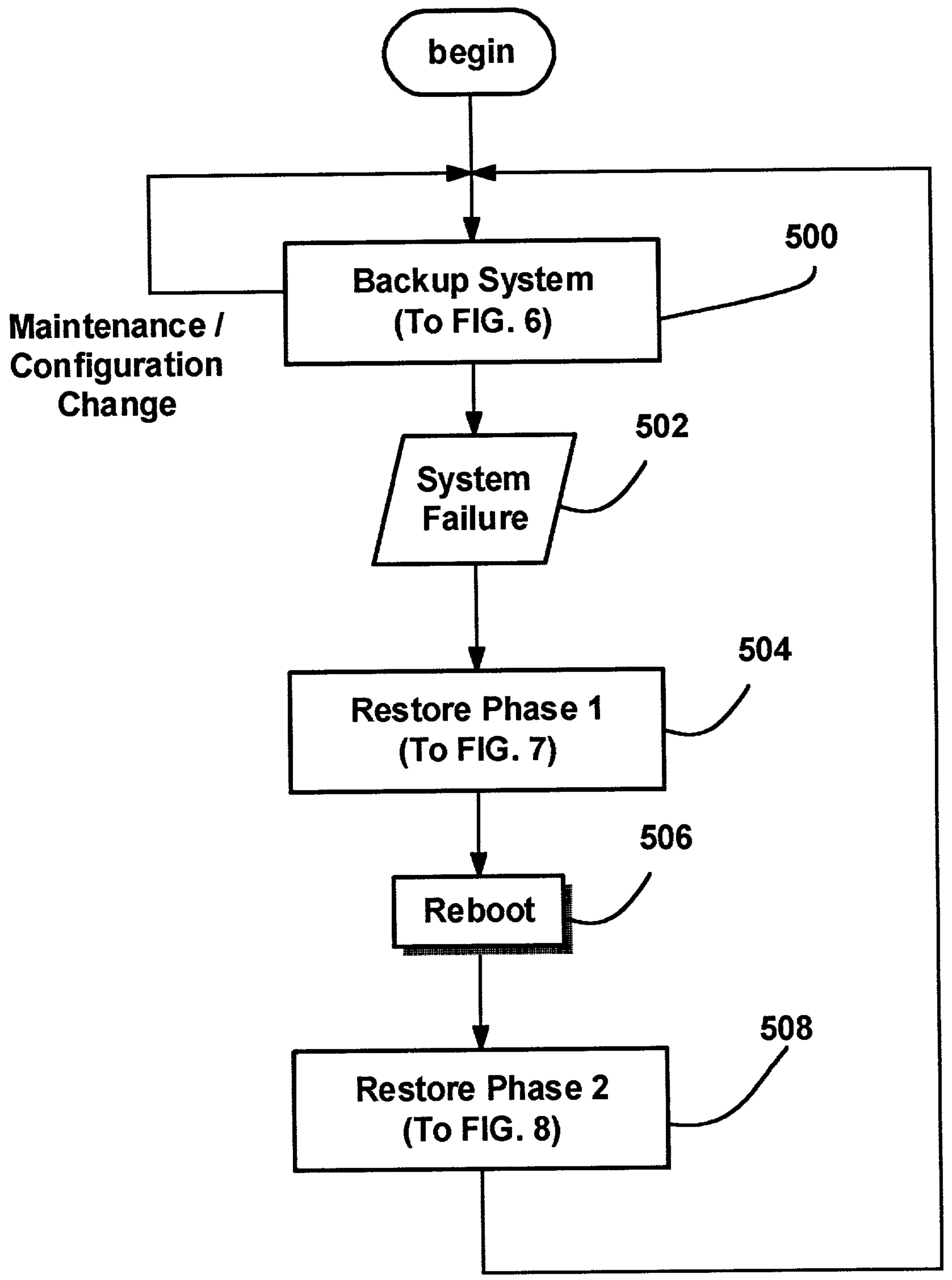

begin
Backup System
(To FIG. 6)
500
Maintenance /
Configuration
Change
System
Failure
502
Restore Phase 1
(To FIG. 7)
504
Reboot
506
Restore Phase 2
(To FIG. 8)
508

FIG. 6

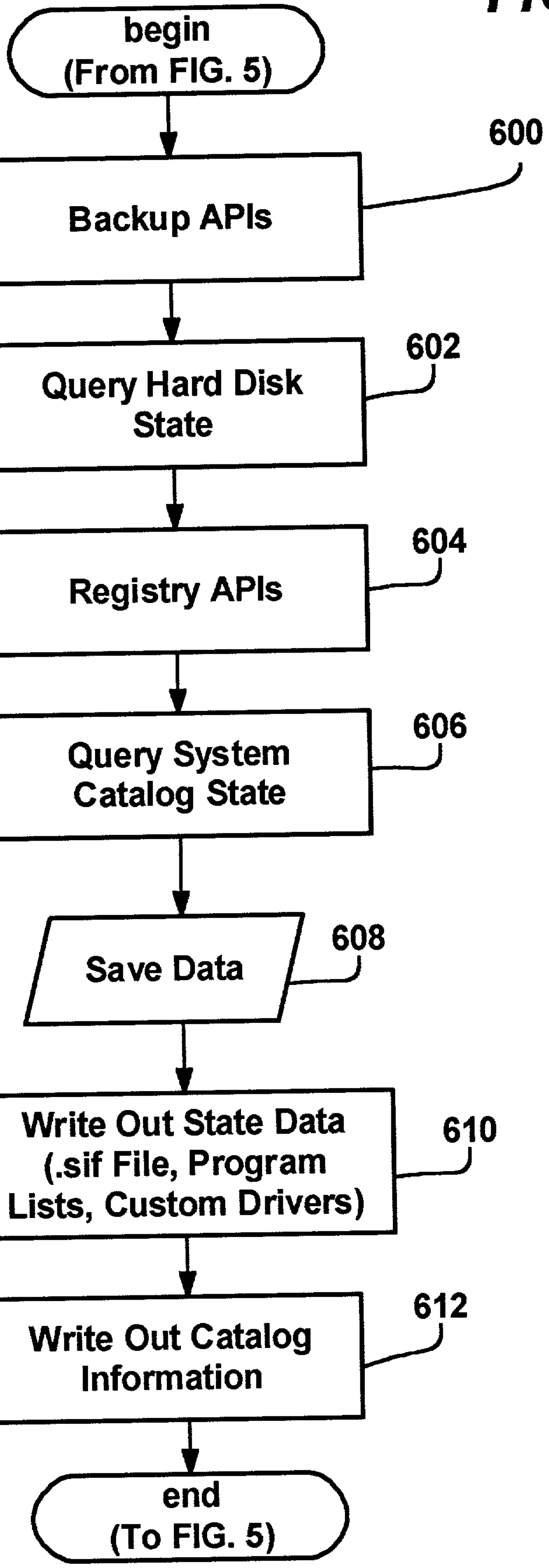

begin
(From FIG. 5)
Backup APIs
600
Query Hard Disk State
602
Registry APIs
604
Query System Catalog State
606
Save Data
608
Write Out State Data (.sif File, Program Lists, Custom Drivers)
610
Write Out Catalog Information
612
end
(To FIG. 5)

FIG. 7

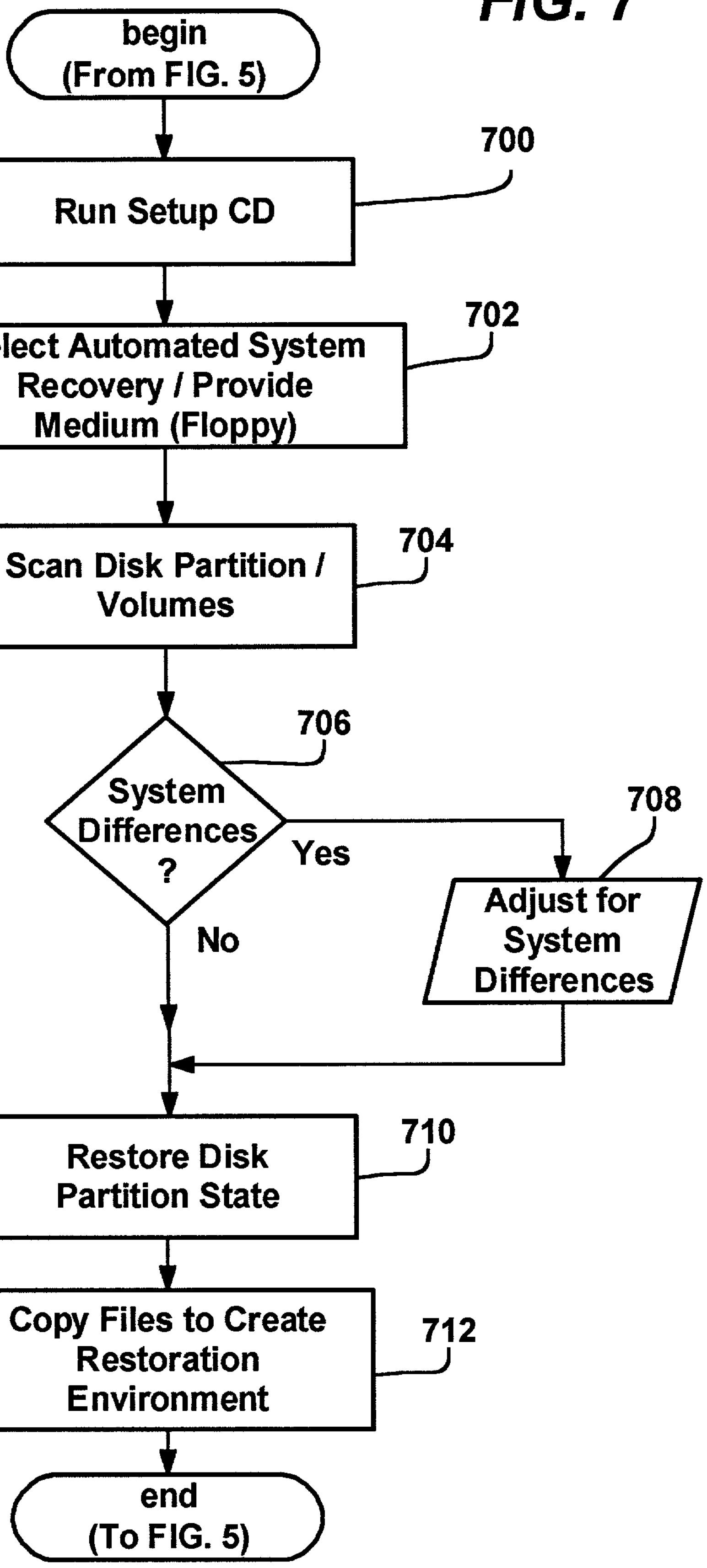
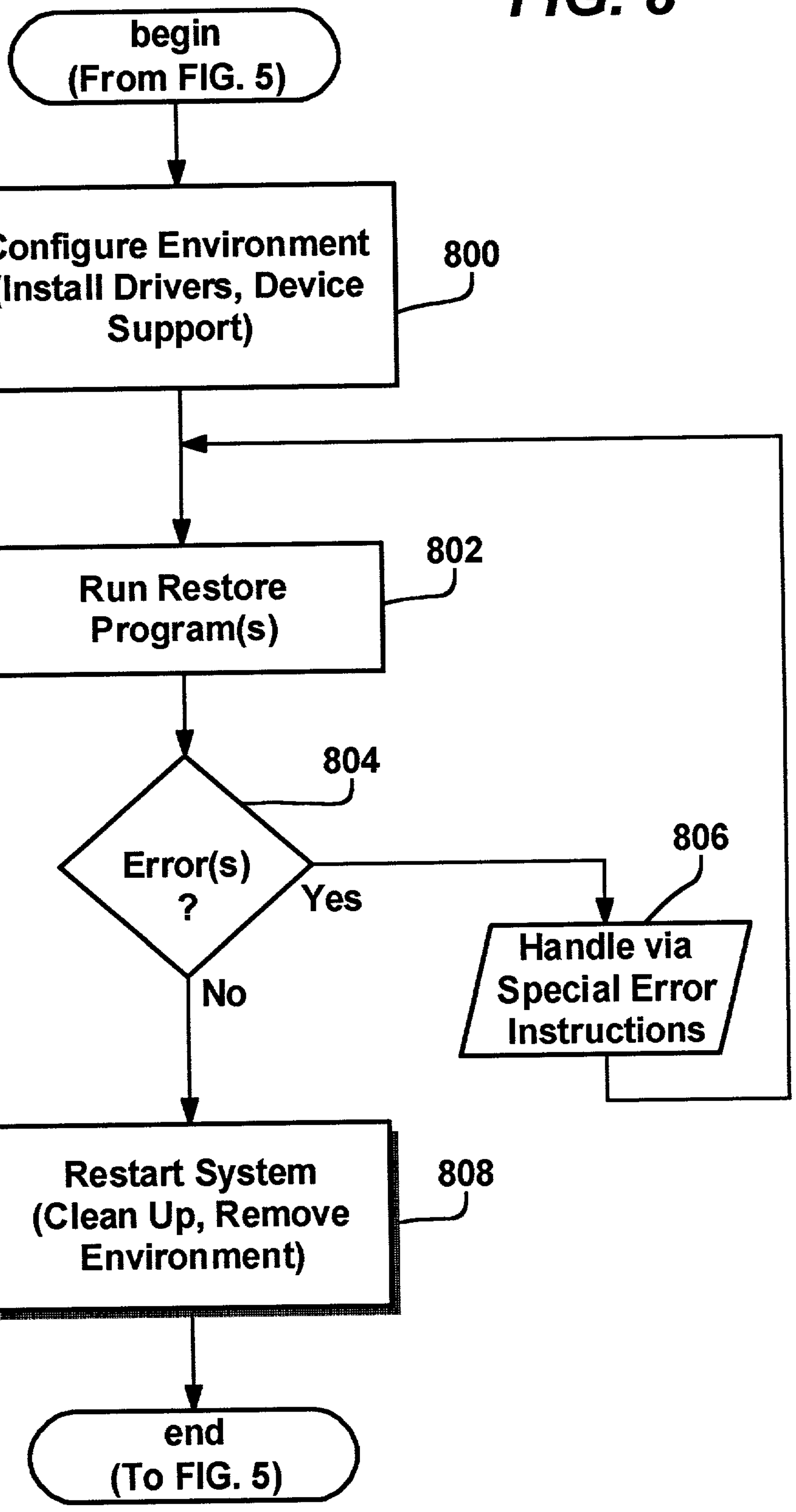

begin
(From FIG. 5)
700
Run Setup CD
702
Select Automated System Recovery / Provide Medium (Floppy)
704
Scan Disk Partition / Volumes
706
System Differences ?
Yes
No
708
Adjust for System Differences
710
Restore Disk Partition State
712
Copy Files to Create Restoration Environment
end
(To FIG. 5)

FIG. 8



begin (From FIG. 5)
Configure Environment (Install Drivers, Device Support)
800
802
Run Restore Program(s)
804
Error(s) ?
Yes
No
806
Handle via Special Error Instructions
808
Restart System (Clean Up, Remove Environment)
end (To FIG. 5)